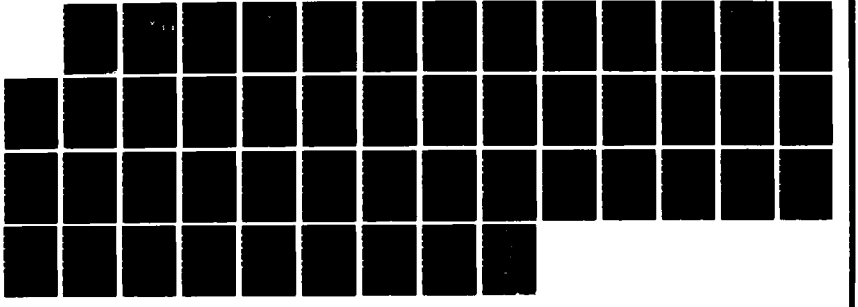
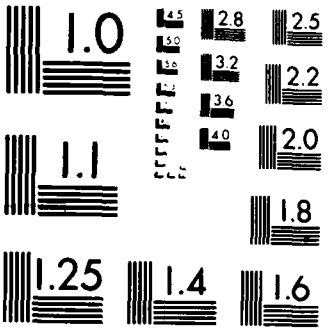
AD-A192 683 SOFTWARE PRODUCT FACTORS FOR DEVELOPMENT COST ESTIMATING AT THE STANDARD SYSTEMS CENTER(U) AIR COMMAND AND STAFF COLL MAXWELL AFB AL R E MEISNER

1/1

UNCLASSIFIED APR 88 ACSC-88-1805 F/G 12/5 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0

AIR COMMAND AND STAFF COLLEGE

STUDENT REPORT

SOFTWARE PRODUCT FACTORS
FOR DEVELOPMENT COST ESTIMATING
AT THE STANDARD SYSTEMS CENTER

MAJOR ROBERT E. MEISNER 88-1805

"insights into tomorrow"

DISCLAIMER

The views and conclusions expressed in this document are those of the author. They are not intended and should not be thought to represent official ideas, attitudes, or policies of any agency of the United States Government. The author has not had special access to official information or ideas and has employed only open-source material available to any writer on this subject.

This document is the property of the United States Government. It is available for distribution to the general public. A loan copy of the document may be obtained from the Air University Interlibrary Loan Service (AUL/LDEX, Maxwell AFB, Alabama, 36112-5564) or the Defense Technical Information Center. Request must include the author's name and complete title of the study.

This document may be reproduced for use in other research reports or educational pursuits contingent upon the following stipulations:

- Reproduction rights do not extend to any copyrighted material that may be contained in the research report.

- All reproduced copies must contain the following credit line: "Reprinted by permission of the Air Command and Staff College."

- All reproduced copies must contain the name(s) of the report's author(s).

- If format modification is necessary to better serve the user's needs, adjustments may be made to this report--this authorization does not extend to copyrighted information or material. The following statement must accompany the modified document: "Adapted from Air Command and Staff College Research Report (number) entitled (title) by (author)."

- This notice must be included with any reproduced or adapted portions of this document.

REPORT NUMBER 88-1805

TITLE SOFTWARE PRODUCT FACTORS FOR DEVELOPMENT COST ESTIMATING AT THE STANDARD SYSTEMS CENTER

AUTHOR(S) MAJOR ROBERT E. MEISNER

FACULTY ADVISOR CAPT RONALD D. FORD, ACSC/XPO

SPONSOR LT COL WILLIAM R. PRICE, HQ SSC/PRS

Submitted to the faculty in partial fulfillment of requirements for graduation.

AIR COMMAND AND STAFF COLLEGE

AIR UNIVERSITY

MAXWELL AFB, AL 36112

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT STATEMENT "A" Approved for public release; Distribution is unlimited.
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) 88-1805	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION ACSC/EDC 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Maxwell AFB AL 36112-5542	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING / SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)

SOFTWARE PRODUCT FACTORS FOR DEVELOPMENT COST ESTIMATING AT THE

12. PERSONAL AUTHOR(S)

MEISNER, ROBERT E., Major, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
	FROM _____ TO _____	1988 April	46

16. SUPPLEMENTARY NOTATION

ITEM 11: STANDARD SYSTEMS CENTER

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The SSC recently adopted a non-proprietary software cost estimating model to support their resource commitments. As with all current models, the SSC model required environmental customizing. The scope of this research was to incorporate relevant product factors into the SSC model. Two factors were selected: product complexity and customer relations. These factors were identified, defined, initialized, and integrated via a theoretical review of SSC needs and industry capabilities.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL ACSC/EDC Maxwell AFB AL 36112-5542	22b. TELEPHONE (Include Area Code) (205) 293-2867 22c. OFFICE SYMBOL

PREFACE

The purpose of this report is to document work commissioned by the Air Force Standard Systems Center (SSC) in identifying realistic product factors for use in their recently adopted software cost estimating model. The mission of the SSC is to develop and maintain Air Force standard computer systems. Like many Air Force organizations, SSC workloads exceed their available resources. Their dilemma is compounded by software cost estimating techniques that are characterized by the Software Engineering Institute (SEI) as generally immature, contributing to an ineffective resource commitment process. The SSC, working with the SEI, is addressing this and other documented problems. Within this larger context, this research project defines two product factors that could significantly affect software development costs at the SSC. The two factors are product complexity and customer relations.

Due to the lack of meaningful historical data, the job of customizing the model became a theoretical selection process to identify, define, and initialize a set of relevant product factors. Once factors were selected, this project suggested a method for incorporating them into cost estimates, and developed an algorithm for adjusting them based on their use in the SSC environment. Throughout the process, this project recognized factor usage based primarily on theory isn't the preferred way to obtain software cost estimates. But, considering the lack of history, it is an acceptable first step.

This research project represents the accomplishments of a group of people. Without their assistance and dedication, the final thoughts and wording of this report would have been much less significant. First, I need to thank Lt. Col. Price, who suggested the problem and kept me on track towards a useful product. In addition, Capt. Mark Dettl, Mr. Dick Stowers, and Mr Bob Tomlin helped me get started with their discussion of the problem and identification of potential sources of information. Their comments on my numerous drafts enhance the utility of this project. Capt. Dettl worked especially hard in critiquing the drafts and coordinating comments for quick turnaround. Thanks is also due TSgt. Pat Brock. His expert advice and timely data base retrievals were critical in assessing the SSC environment. Finally, Capt. Ron Ford thought-provoking questions were always on target and provided an independent level of technical

expertise to the final product. All those mentioned are true professionals.

The list of helpers wouldn't be complete without mention of the contributions of my wife. Wanda's understanding and support were critical to the quality of this project. The number of times she reviewed the drafts for administration and comprehension should qualify her for college credits in software engineering. As a new bride, her discipline in locking me in the study on nights and weekends far exceeded what anyone could reasonably expect, or what I cared to endure. Gulf Coast beaches here we come!

ABOUT THE AUTHOR

Major Bob Meisner has had a varied career in the computer field. He has held jobs as a systems analyst in support of Automatic Digital Network (AUTODIN) terminals at the US Army Communications Command, Single Integrated Operational Plan (SIOP) production at the Strategic Air Command, and communications systems at the Air Force Weapons Laboratory. His last 22 months at the Weapons Lab were spent as Chief of the Computer Operations Division, directing operations of the USAF's first Cray supercomputer. Prior to attending Air Command and Staff College, Major Meisner served for two years as the Information Systems Inspector for the Air Force Systems Command Inspector General. His next assignment will be as Commander of the 1855th Command, Control and Intelligence Support Squadron at Osan, Korea.

Major Meisner earned a Bachelor of Science degree in Computer Science from Northeast Louisiana University in 1974. Six years later he entered the Air Force Institute of Technology, School of Engineering, graduating with a Master of Science degree in Computer Technology. His technical education continued in 1983 with his attendance at the Computer Systems Staff Officer Course. Major Meisner's professional military education includes attendance at the Army Signal Officers Basic Course, Squadron Officers School, and Air Command and Staff College; and completion of the Marine Corps Command and Staff College by correspondence.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

TABLES

GLOSSARY

CAMPUS	Center Automated Manpower and Project Update System
COCOMO	Constructive Cost Model
IEM	Ideal Effort Multiplier
IEM(P,PF)	Ideal Effort Multiplier Function
MM(P,PF)	Man-month Function
P	Project (used only in IEM and MM functions)
PF	Product Factor (used only in IEM and MM functions)
PR	Productivity Range
SDC	Systems Development Corporation
SEI	Software Engineering Institute, Carnegie-Mellon University
SEL	Software Engineering Laboratory, University of Maryland
SLOC	Source Lines of Code
SSC	Standard Systems Center
WBS	Work Breakdown Structure

EXECUTIVE SUMMARY

Part of our College mission is distribution of the students' problem solving products to DoD sponsors and other interested agencies to enhance insight into contemporary, defense related issues. While the College has accepted this product as meeting academic requirements for graduation, the views and opinions expressed or implied are solely those of the author and should not be construed as carrying official sanction.

"insights into tomorrow"

REPORT NUMBER 88-1805

AUTHOR(S) MAJOR ROBERT E. MEISNER, USAF

TITLE SOFTWARE PRODUCT FACTORS FOR DEVELOPMENT COST ESTIMATING AT THE STANDARD SYSTEMS CENTER

I. Purpose: To identify realistic product factors for use in Standard Systems Center's (SSC) recently adopted software cost estimating model.

II. Problem: The mission of the SSC is to develop and maintain Air Force standard computer systems. Like many Air Force organizations, SSC workloads exceed their available resources. Their dilemma is compounded by software cost estimating techniques that were characterized by the February 1987, Software Engineering Institute (SEI) Review of the SSC as generally immature, contributing to an ineffective resource commitment process. The SSC, working with the SEI, is addressing this and other documented problems. Within this larger context, the problem tackled by this research was to identify, define, initialize, and integrate a set of realistic product factors into the newly adopted SSC model.

III. Discussion: Authorities describe over a hundred factors that influence software costs. They generally categorize these factors by four aspects that characterize a software development environment - people, process, computer, and product. This

research deals only with those factors that portray the problem being automated, that is, the product factors.

Due to a lack of meaningful historical data, the job of customizing the factors for use in the model became a theoretical selection process to identify, define, and initialize a set of relevant product factors. Throughout the process, this project recognized that factor usage based primarily on theory wasn't the preferred way to obtain software cost estimates. But, considering the lack of history, it was an acceptable first step.

Once the need for product factors was established, five criteria were developed to guide the factor customizing process. The criteria were scope, significance, measurability, ease of use, and factor independence. The customizing process began by selecting two significant product factors out of the universe of potential factors. A recent study by Capers Jones and the more widely analyzed studies of IBM-FSD, ITT, COCOMO, and the Software Engineering Laboratory of the University of Maryland were used to identify product complexity and customer relations as the top two significant factors. Following their selection, the two factors were substantively defined. The product complexity definition was essentially copied from Barry Boehm's COCOMO. The customer relations definition was original, although suggestions of its various aspects were found scattered throughout the literature. Finally the two factors were initialized. Initial values resulted from analyzing the five studies listed above. This analysis chose the middle values from the studies, as a hedge against outrageous estimates until the factors can be historically calibrated to the SSC environment. Product complexity settled on the values of COCOMO, while customer relations used those of the ITT study.

Integration of the two factors into the SSC model involved describing how they could be used and calibrated. Factor values are effort multipliers of source lines of code (SLOC) baseline estimates. The baseline estimates reflect the actual number of SLOC required to automate a task. The multipliers scale this number to reflect variance in the levels of effort indicated by the factors. Product factors don't change the actual number of SLOC in a delivered product, but do account for product complexity and customer relations in productivity and cost estimates. After the factors have been used, and a history built, they will likely require calibration to increase the accuracy SSC cost estimates. The method of Ideal Effort Multipliers (IEM) developed by Barry Boehm was adapted for calibrating the SSC model. This research describes how to use the IEM to adjust not only the product factors, but also other factors used in the model.

IV. Recommendations: This research report makes two recommendations. One is that the customer relations factor be refined before being put into use. The second is that the two product factors be incorporated into the current SSC software cost estimating model. While product complexity is explicitly defined and used in COCOMO, customer relations is discussed only generically in the literature. As a result, the definitions of customer relations listed in this research are original and haven't been validated in an operational environment. Hence, the first recommendation follows. The second recommendation results from authoritative studies which consistently show that product complexity and customer relations impact software costs.

Chapter One

INTRODUCTION

The mission of the Standard Systems Center (SSC) is to develop and maintain Air Force standard computer systems. Like many Air Force organizations, SSC workloads exceed their available resources. Their dilemma is compounded by software cost estimating techniques that are characterized by the Software Engineering Institute (SEI) as generally immature, contributing to an ineffective resource commitment process [18:4]. The SSC, working with the SEI, is addressing this and other documented problems. Within this larger context, this research project will define how product factors affect software development costs at the SSC.

BACKGROUND

In February 1987 the SEI conducted a management review of the SSC's software engineering practices. The review was conducted under the rules outlined in the Assessment Agreement signed on 12 February 1987. The agreement was well-planned and resulted in 10 major recommendations for improved operations. These recommendations were published in the SEI's "Final Report and Recommendations, Software Engineering Institute Review, Standard Systems Center, Gunter Air Force Station, 17-19 Feb 87" [18].

Following the publication of the final report, the SSC developed an action plan, "Response to Software Engineering Institute Findings and Recommendations." In their action plan the SSC acknowledged the SEI recommendations and set out to implement them [20]. They acted on the first recommendation, Project Management, by adopting the SEI framework for software project planning, which included a model for cost estimating. The SEI model recognized that many factors contribute to software costs, but left quantification of them to the SSC. Using the SEI model as a basis, the SSC is defining a model for their specific use. As part of the definition process, the SSC commissioned this research project to identify and quantify product factors that could affect software costs at the SSC.

PROBLEM STATEMENT

The purpose of this project is to develop a system for incorporating realistic product factors into cost estimates at the SSC. Attaining this purpose requires the achievement of four objectives. The first is to define the need for product factors in software cost estimates. The second is to identify and quantify a set of realistic factors that influence development costs at the SSC. The third is to develop a method for incorporating these factors into cost estimates. The last objective is to develop a method for adjusting the factors based on SSC experience gained through their use.

ASSUMPTIONS AND SCOPE

The four objectives listed above are limited in scope by three assumptions. The first assumption is that product factors identified by this research project will be used in the SSC software cost estimating model. The second is that productivity factors, other than those derived from product attributes, will be defined and validated by another process. The last is that factor values are initial, educated estimates which will probably require adjustments.

Note that this report refers to productivity and product factors almost interchangeably. They aren't interchangeable. Product factors are those that describe the problem being tackled, and are a subset of the more general class of attributes known as productivity factors. People, process, and computer attributes are other factors which affect productivity. The literature doesn't restrict itself to analysis of only product factors, but studies productivity factors in general. So, literature reviews generally cite productivity factors, while specific applications specify product factors. Further details of their relationship can be found in Chapter 2, The SSC Model.

These three assumptions and the fact that the SSC cost model was already in preliminary use dictated the following scopes for the four objectives listed in Problem Statement. The first objective was satisfied through a review of current literature and analysis of the SSC software cost model. The second was attained through reviews of current literature and SSC historical documents. The last two objectives were accomplished by incorporating the results of the second objective into the cost estimating system used by the SSC.

SEQUENCE OF PRESENTATION

The general flow of this report follows the objectives listed under Problem Statement. First, the need for product factors in estimating SSC software costs is defined in Chapter 2. Chapter 3 identifies a realistic set of factors and establishes initial values for their use. Then, Chapter 4 describes how the factors from Chapter 3 are used and calibrated with the SSC model. Finally, Chapter 5 analyzes how well the chosen factors satisfy the criteria listed in Chapter 2, and makes recommendations for use of the factors developed by this project.

Chapter Two

THE NEED FOR SOFTWARE PRODUCT FACTORS

> False scheduling to match the patron's desired date is much more common in our discipline than elsewhere in engineering. It is very difficult to make a vigorous, plausible, and job-risking defense of an estimate that is derived by no quantitative method, supported by little data, and certified chiefly by the hunches of the managers.

While this quotation appeared in 1975 in The Mythical Man-Month, it accurately summarizes the project management finding in the SEI final report. Frederick Brooks' proposed solution in 1975, like the SEI recommendation in 1987, called for developing estimation rules and productivity factors [6:21; 18:4,7]. Since the SEI review, the SSC has inaugurated a cost estimation model. However, the product factors, a subset of productivity factors, used in the model haven't been established [19:5-6]. This chapter will define the need for these product factors by reviewing the state of the art of software cost estimating. In addition, this chapter will describe an approach for defining product factors by explaining the SSC model and specifying criteria for applying factors at the SSC.

THE STATE OF SOFTWARE COST ESTIMATING

The credibility of software development organizations rests on the immature science of software engineering economics. Without reliable software cost estimates, projects have no firm basis for defending unusually optimistic budgets and tradeoff analyses, and often compromise on performance to meet schedule overruns [4:30]. Unfortunately, software cost estimating isn't yet a robust science, due primarily to the lack of suitable measurements for programming attributes [9:47,51; 11:393; 14:2]. This dependable, cost estimating dilemma isn't surprising given the recent, meteoric ascendancy of the software industry.

Electronic computing is an infant technology, born only as recently as 1930. From its birth it took 16 years until the first general-purpose electronic computer, the ENIAC (Electronic Numerical Integrator and Calculator), commenced operation.

Still, it wasn't until 1951 that the first commercial computer was delivered, the UNIVAC I (Universal Automatic Computer). The computer revolution had dawned [12:18,22].

In the early days of the revolution software contributed little to the costs of a fully functioning computer. Greater than 80 percent of the costs were attributable to hardware. However, as the revolution gained momentum, software began to dominate computer system costs. Boehm predicted that the relative costs flip-flopped around 1965 [3:326]. It has only been since then that the need for software cost estimating has become a significant problem.

Yet, it wasn't until a silent coup in the mid-1970s that the software gurus became the new leaders of the revolution. It was at this time that the classic essays of Brooks and Boehm were published. In 1975 Brooks called for better software cost and productivity estimating techniques [6:21]. In 1976 Boehm suggested that hardware engineering was a much more advanced scientific discipline when compared to software engineering. Further, he stated that "the most pressing software development problems are in ... requirements analysis design, test, and maintenance of applications software by technicians in an economics-driven context" [3:352].

Over 10 years later software cost estimating is still an immature discipline. While the problems associated with estimating are better defined, solutions aren't yet fully developed. Several authors suggest this immaturity is a consequence of poor programming measurement techniques [9:47,51; 11:393; 14:2]. Jones goes one step further to identify five underdeveloped areas where progress is being made. Especially encouraging is his prognosis for notable results in economic productivity metrics by 1990 [15:43,83].

Even though software cost estimating is a maturing science, the utility of cost models is well-established. A literature review identified a plethora of cost models, both proprietary and non-proprietary, each with inherent strengths and weaknesses. Strengths are particularly pronounced in the environments under which the models were developed and provide less accurate estimates in other environments. This characteristic of tight coupling restricts the portability of models to new environments. All models recognize this trait and compensate for it by supplying utilities to customize each to a new environment. Still, no model accurately captures the product and environmental attributes which affect resource expenditures [1:107; 7:27; 13:8; 6:427].

However, software cost estimation models are beneficial for producing product plans. The SSC recognized this fact and

accepted the SEI recommended model outlined in the extract from Humphrey's book [13] (this model is referred to as the SSC model throughout this research report). The SEI model is based on source lines of code (SLOC) with adjustments for goodness of estimate and productivity factors, including product factors. Like any other model, these adjustments must be customized and calibrated to the SSC [19:5-4 - 5-6]. The purpose of this research project is to customize those factors related to product variations.

THE SSC MODEL

Before the customization process can proceed, the nature of the SSC model must be identified. This identification allows product factors from other studies to be applied in proper context. In the vernacular of Basili, the SSC model is an "adjusted baseline, static, multivariable" model. That is, the fact that the model combines system size with contingency and productivity factors to determine a single effort value makes the model multivariable and static. Further, since the contingency and productivity factors are used to scale the estimate of system size, the model is referred to as adjusted baseline [2:4-5].

Exploring the mechanics of the SSC model will help justify these classifications. The model's baseline is established by estimating the SLOC needed to satisfy user requirements. Then, the baseline is adjusted by multiplying it by contingency and product factors. The resulting product is divided by the base average programmer productivity to yield an effort value. While this value is sufficient for the purposes of this research, the final step in obtaining a cost estimate is to apply budget data to the effort value [19:5-1 - 5-9].

In practice, the estimation process outlined above is accomplished through the use of a work breakdown structure (WBS). That is, baseline SLOC estimates and scaling factors are applied to components of a WBS, not the composite system. The final effort value is the sum of the component estimates. Also, the order of adjustments to the baseline is different, in practice, than outlined above. But, the result is effectively the same.

With a basic understanding of the SSC model, the scope of the productivity and product factors can be put into context. Studies have shown cost attributes that impact software productivity fall into four categories as outlined below [4:115-116; 7:238]. For the purposes of this research project the aggregation of attributes is referred to as productivity factors.

> People factors relate to the capabilities of programmers working on the project. Examples include

individual expertise, team organization, morale, and quality of management.

Process factors describe the production environment. Examples include programming language, structured programming techniques, software tools, and data base availability.

Computer factors deal with development computer attributes. Examples include response time, system volatility, and turnaround times.

Product factors portray the problem being tackled. Examples include product size, state of problem definition, level of documentation, solution complexity, reliability, and security restrictions.

As stated in Chapter 1, Assumptions and Scope, this research project will deal only with applicable product factors and will assume the other factors have been suitably incorporated into the model. This assumption is realistic. In effect factors which remain relatively constant from project to project appear as integral parts of the baseline and average programmer productivity estimates. By implication these include such attributes as individual expertise, organizational management, programming language, software tools, system response time, and system turnaround times. Minor fluctuations in these attributes are compensated for at the lowest organizational level where the estimating is done, the software is developed, and where the attributes are relatively static. Consequently, this research examines only the effects which the problem to be solved contributes to the cost estimating process, that is, product factors.

CRITERIA FOR PRODUCT FACTORS

For this project, the phrase "customizing the product factors" involves identifying and initializing those factors that apply to the general SSC software development environment. As alluded to earlier, software cost estimating and productivity measurement is an immature science. Consequently, there is no definitive set of factors from which to choose a subset that applies to any particular environment. Three widely analyzed models illustrate this point. The first popular study was completed by the System Development Corporation (SDC) in 1966 and identifies 104 significant productivity factors. Another classic study, completed by Walston and Felix at IBM-FSD in 1977, lists 29 cost attributes. One of the most popular models, Boehm's Constructive Cost Model (COCOMO), incorporates 15 productivity factors [4:114,512,517]. A fourth source of factors is the

study by Jones which outlines 20 major and 25 potential factors [15:85]. While some factors overlap between the studies, no definitive set of factors emerges. Such a wide variety complicates the customizing task. Therefore, the following five subsections describe the criteria that will be used in selecting the product factors applicable to the SSC.

Scope

Boehm defines scope as the class of software projects whose costs need to be estimated [4:520]. The implication of such a definition on this research is that the class of projects chosen determines how well product factors influence cost estimates. The wide variety of factors discussed in technical fora leads to the hypothesis that some factors will have more applicability to the SSC than others. Identifying the scope at the SSC helps focus the search for relevant factors.

The first recommendation from the SEI Review was that the SSC develop a project management process that includes estimating product size, costs, and schedules. The SEI Review found that the SSC didn't have an effective commitment process for development efforts. This ineffectiveness was manifest in the SSC's frequent acceptance of customer delivery demands without first sketching a development plan. Consequently, the impact of new requirements couldn't be assessed, and the SSC had no basis for scheduling work [18:4,7]. Institutional use of the SSC cost estimating model is designed to address the size and cost aspects of the project management process, and contributes to realistic scheduling. Within this context, the scope includes production and modification developments, and excludes maintenance.

Many aspects of software development, such as design and testing, contribute to costs without increasing the number of SLOC developed. These relationships are captured in the factors used to adjust the baseline SLOC estimates as described in The SSC Model. The SSC chooses to include the software definition, design, development, and testing phases in their cost estimates. Therefore, product factors must account for these aspects.

Boehm, Conte, and Jones all express caution in using SLOC as a unit of measurement. Yet, all concede that while other units may prove to be more accurate in the future, SLOC is sufficient for current estimates. However, a clear definition of what is included in counts of SLOC is paramount when comparing data from different organizations [4:58-59,477-482; 7:236-237; 15:81-82,90]. The SSC includes all lines, except comment and blank lines, which are input into a compiler and are required to complete applicable tasks [8:--]. This research will do likewise.

In summary, product factors and their initial values must accommodate the following aspects of the software projects being estimated. First, factors must only account for production and modification tasks. Also, they must describe the definition, design, develop, and test phases of the development cycle. Finally, SLOC comparisons must include all code required to complete applicable tasks, excluding comments and blank lines.

Significance

As Aristotle wrote, "It is the mark of an instructed mind to rest satisfied with the degree of precision which the nature of the subject admits and not to seek exactness when only an approximation of the truth is possible." This thought is a basic tenet for the selection of factors to be applied at the SSC. Further, it implies that a minimum set of significant factors might be sufficient for estimating costs. This implication has been supported by several authorities. One of Boehm's ten criteria for evaluating a software cost model is parsimony. That is, does the model avoid factors that don't contribute appreciably to the results [4:520]? Bailey and Basili's search for a meta-model at the Software Engineering Laboratory (SEL) at the University of Maryland concluded that too few productivity attributes mean that a model will miss potentially significant variations, while too many make the analysis meaningless. But, they go on to suggest that missing some variations don't "weaken the accuracy of the model beyond useful proportions" [1:115].

Further, Humphrey believes that productivity factors are important in improving the estimates within particular organizations. However, he cautions against becoming more sophisticated than the available data warrants. He hypothesizes that SLOC estimates, the basis for the SSC model, are usually wrong by 100 percent. Further, even ideal productivity factors are rough estimates. Consequently, only one or two significant factors are warranted during the initial use of a model. Sophistication in the form of additional factors can be added only after a large database of information has been accumulated and analyzed [13:4-5].

Conte goes further by suggesting that only factors accounting for a significant percentage of variance should be included in a model. As an aid to determining significant factors, Conte used the concept of productivity range (PR). Each product factor has a variable effect on the cost estimate according to the level of severity which the factor imposes. For example, problems with stringent timing requirements may increase SLOC estimates by 2.5 times, whereas simple mathematical problems might decrease estimates to .9 times. These hypothetical multipliers indicate lower productivity and higher costs for timing over mathematical

problems. The PR is the ratio of the largest to the smallest value for a factor. In the example above, the PR indicates productivity on mathematical problems is 278 percent (2.5 / .9) greater than on timing problems. The PR is useful in identifying factors with the most extensive effects on cost estimates [7:239,246].

Considering the infant state of the SSC model, this research project adopts Humphrey's suggestion and will identify two significant product factors. Additional factors can be added following operational use of the model. Also, to be classed as significant, product factors must have at least a PR of 200 percent.

Measurability

The utility of a model results from the consistency of its estimates. Ideally, when a single project is independently estimated by two people using the same model within the same organization, the results are identical and equal actual costs. While attaining the ideal is unlikely, it is approachable only by minimizing the subjectivity of the estimators. Consequently, the application of individual product factors must be objective and quantifiable [4:520-522; 7:239].

Ease of Use

In general, the utility of any system is ultimately judged on the ratio of input effort to output results. Similarly, a cost estimating model, which is newly introduced to an organization, will initially be judged on the ease with which it can be understood and with which data can be input. Later, outputs from the system will contribute to perceptions of its utility. So, product factors must be well-defined, significant and easy to obtain [4:520,523].

Factor Independence

The reason for including a criterion for independence is primarily mathematical. But, ease of use is an important benefit. Mathematically, a single, composite product factor is the product of multiple, independent factors. Such factors tend to be easily understood and used since there is no complex relationship between factors.

Boehm and Conte go further to suggest that highly correlated factors should be combined into one [4:523; 7:239]. For example, an organization might employ structured programming, chief programmer team techniques, and code inspections in their software development. While each could affect cost estimates, it

isn't clear they are independent. For practical purposes they could be combined into one factor called structured programming.

In summary, product factors must be independent. This criterion simplifies the mathematics used in the cost estimating model. Also, it makes the model easier to understand and use.

Chapter 3

SELECTING THE PRODUCT FACTORS

Numerous studies have defined a myriad of productivity factors in attempts to capture the effort required in software development. Boehm discusses nine in his book [4:510-520], Conte reviews several more [7:234-272], and Jones provides some recent new additions [15:85-245]. While these studies showed a good correlation between the factors chosen and the sample programs used to build and validate the models, portability of factors to other environments is poor. Authorities caution against using productivity factors without calibration to the environment where they will be used [1:107; 4:140; 7:27; 13:5].

In some situations, such as those at the SSC, history is inadequate to customize factors. In these cases a theoretical model can be useful. All of the models referenced in this study (see Table 2) were born of theory, and calibrated and proven in specific environments. The SSC chose not to develop a totally new model, but built on the lessons learned from past studies. The SSC approach of adopting a model, assigning feasible, theoretical initial values to adjustment factors, and calibrating them based on empirical evidence is valid [7:114; 19:5-9]. This research supports the SSC's initiation of a customized estimating model by selecting, defining, and initializing product factors.

THE SSC ENVIRONMENT

Before applicable factors can be effectively employed, a basic understanding of the target environment is necessary. The SSC Center Automated Manpower and Project Update System (CAMPUS) provides the historical data used to characterize the SSC software development environment. It was chosen for the task because it is the only consolidated repository of management information maintained for all production projects at the SSC. The data summarized by CAMPUS isn't analyzed in this section. It is merely displayed along with a discussion of its constraints. Data interpretation is deferred to the factor selection, definition, and initialization sections of this report.

The following definitions from SSCR 700-7, Vol I, Center Automated Manpower and Project Update System, clarify the terminology used by the CAMPUS data base [21:5-5,A8-2].

> Production - time expended developing new applications, including requirements analysis, design, programming, documentation, testing, and release of systems.
>
> Modification - time expended altering an existing system to support constantly evolving requirements.
>
> Maintenance - time expended eliminating faults to ensure that applications are working properly.
>
> Complexity factor - project attributes used for maintaining the manpower standards established for directorates. CAMPUS uses the attributes to calculate a complexity point total for each project.

The CAMPUS data summary shown in Table 1, SSC Complexity Factor Breakout, gives an indication of the relative usage of the listed attributes during the time periods shown. The attribute summaries include only production and modification projects. The data base contained 2687 entries for the 1978 to 1987 time frame and 1581 for 1986 and 1987 [5:--]. The two time periods are listed to identify any recent trend changes. Attributes comprising five percent, or less, of the listed factor are summed under the title "Other". The attributes listed under "Interfaces" are all for functional software. In practice more than one attribute may apply to a single factor. However, CAMPUS restricts inputs to only one per factor. Further, the default attribute for each factor is "Not Applicable" [21:5-5].

CAMPUS has copious other data in its data base. Some data fields that would appear applicable to this research aren't used. While a SLOC field is available, it hasn't been used over the years. On the other hand, man-hour estimates are input, but are allowed to change as projects progress. Consequently, the utility of these two essential factors are ineffective for selecting and calibrating factors.

Experts recommend the use of historical data to calibrate an estimating model [1:107; 4:524; 7:27]. By itself, the data in CAMPUS is inadequate for selecting and initializing product factors. However, it does highlight the environment where the SSC model must perform. Consequently, its primary utility lies in customizing definitions after the factors are selected and before they are initialized. For example, suppose program complexity was selected as a factor for use in the SSC model. CAMPUS could be used to determine whether classified data should be included in the definition of program complexity.

Factor	Attribute	% of Projects 78-87	% of Projects 86-87
Inter-faces	Not Applicable	33.3	33.7
	w/i ADPS-w/i Division	13.1	11.5
	w/i ADPS-across Directorate	15.6	20.9
	Between ADPS - Outside SSC	20.1	19.1
	Other (7 categories)	17.9	14.7
Hardware	Not Applicable	86.5	87.5
	On Contract	9.2	9.8
	Other (2 categories)	4.3	2.6
Privacy/ Security	Not Applicable	77.5	83.2
	Privacy Act	18.3	15.2
	Other (2 categories)	4.2	1.6
Program Language	Not Applicable	27.7	32.4
	COBOL/Fortran/Other	67.4	66.3
	Other (3 categories)	4.9	1.3
Customer Range	Not Applicable	27.5	32.2
	Single Air Staff OPR (Base Level)	35.0	41.2
	Multi Air Staff OPR (Base Level)	23.7	16.8
	Other (3 categories)	13.8	9.8
Input Method	Not Applicable	27.8	32.4
	Tape/Disk/Teletype/RCR	41.5	41.3
	TC 521/CRT/Key to Disk	8.9	9.7
	Frames/Cassettes	16.5	15.1
	Other (2 categories)	5.4	1.5
Output Method	Not Applicable	27.8	32.4
	Paper/Cards	7.2	3.3
	Tape/Disk/Teletype/Terminal Point	46.0	48.5
	CRT/RLP/TC 521	7.9	6.8
	Frames/Cassettes	10.6	8.5
	Other (1 category)	.6	.6
Type System	Not Applicable	27.2	32.2
	Batch	22.6	12.4
	On-line	21.2	19.1
	Interactive	29.0	36.3
Data Base	Not Applicable	32.5	32.9
	Simple	11.6	8.6
	Complex	10.7	7.0
	Very Complex	45.3	51.5

Table 1. SSC Complexity Factor Breakout.

FACTOR SELECTION

A recent study by Jones and the more widely analyzed studies of IBM-FSD, SEL, ITT, and COCOMO are used to identify the top two product factors as suggested by the significance criterion from Chapter 2. Table 2, Potential Product Factors, summarizes the product factors listed in each of the studies which falls within the scope of this research project. With one exception, all applicable factors from the studies are listed. The IBM-FSD list shows only the top nine factors, culled by PR. While each study produced favorable results in their respective environments, each cautioned that the resultant model would require calibration for use in new organizations. The factor selection process discounts the specific values assigned in the studies. Instead, it concentrates on relative magnitudes in order to identify the largest consistent contributors to software costs.

A factor contributing to subjectiveness in the selection process is that the studies didn't use a common data recording technique. The following study synopses illustrate some of the differences.

> The IBM-FSD study assigned a mean productivity on a two or three point scale in units of delivered SLOC per man-month. For example, customer interface complexity is assigned one of three values based on an assessment of less, equal, or greater than normal complexity. The 29 factors listed in the study were analyzed independently [23:63-65]. Actually, the factors aren't independent and interact in complex ways [4:517; 7:246].
>
> The SEL study subjectively identified significant product factors, but didn't assign any particular values. For simplicity each factor was assigned the same range of values. In practice, these factors would be scaled by coefficients obtained empirically from the environment where the model is used. Further, the study didn't attempt to compensate for the correlation between factors [1:111-115].
>
> The ITT study narrowed 100 productivity factors down to nine significant factors, which were "substantially associated with productivity [and] distinguishable from other performance factors." With the exception of product size, which wasn't rated, factors are measured on a three point scale in units of percentage of variation from a mean productivity value. These values aren't explicitly listed, but are published graphically [22:144-150].

Factor	PR
IBM-FSD (Walston-Felix)	
Customer interface complexity	4.03
User participation in requirements definition	2.40
Customer originated program design changes	2.94
Customer experience with application area	2.84
Percentage of programmers doing development who took part in functional specifications	2.56
Complexity of application processing	2.08
Program design constraints on main storage	2.04
Percent of code delivery	2.06
Pages of delivered documentation per 1000 delivered lines of code	2.56
SEL (Bailey-Basili)	
Customer interface complexity	
Customer initiated design changes	
Application process complexity	
Program flow complexity	
Data base complexity	
ITT	
Resource constraints	1.6
Program complexity	1.6
Client interface	2.7
Size of programming product	n/a
COCOMO (Boehm)	
Required software reliability	2.00
Data base size	1.33
Product complexity	2.36
Required development schedule	1.09
Jones	
Program size	1.98
Complexity of program and data	2.90
Program type	1.38
Enhancing existing programs	1.62
Documentation	1.33

Table 2. Potential Product Factors.

The COCOMO model chose cost driver factors for their significance to general situations and their independence from product size. Factors are measured on a four, five, or six point scale in units of percentage of variation from a nominal productivity value. The model appears continuous through its suggestion to interpolate values between points on the scale [4:115-118,132-133]. While factors may be independent of product size, some appear highly correlated with each other [7:305].

The Jones' factors were identified and initialized in isolation from each other. Factor values were determined from case studies which presumably represent the class of problems they portray. Factors are measured on a three or four point scale, generally in units of SLOC per month. Factors aren't independent [15:85-86].

Regardless of the study differences, the data in Table 2 is sufficient to select the largest contributors to software costs. Four of the five studies have adequate data upon which to calculate PRs. In the IBM-FSD and Jones studies, the PRs are computed by dividing the larger value for SLOC by the smaller. The ITT and COCOMO factors are published as percentages from a mean number of SLOC. Their PRs are the ratio of high percentage to low. Mathematically, these PR values are equivalent and yield valid comparisons.

Each of the studies with a calculated PR has one factor that stands out. In the IBM-FSD and ITT studies, the predominant factor by almost a power of two is customer interface complexity. The COCOMO and Jones studies indicate that product complexity is the dominant factor. Additional analyses support the selection of these two factors.

Product complexity imposes a productivity range of greater than two on the four models with PR values. Also, three of the five factors identified as "significant" by the SEL study are directly related to product complexity. Further, the various aspects of product complexity are some of the most widely analyzed topics in the software industry. Articles abound by authorities other than those cited in this research [15:68-73].

Customer interface complexity is also considered significant by all of the listed studies. While the SEL study didn't assign PR values, two of its five factors apply to customer relations. Similarly, the three largest PRs in the IBM-FSD study involve customers. Even though the COCOMO and Jones studies considered customer relations to have a significant effect on software costs, they didn't include such a factor for different reasons.

While Boehm found that customer interface quality could double costs, he attributed this to poor management and considered incorporation of such a factor as a disincentive to efficiency. He speculated that a low productivity estimate based on poor management factors would be a self-fulfilling prophecy [4:487-488]. Jones, on the other hand, included user participation in his list of 25 "known to be significant" factors that lack sufficient data for valid quantitative assessments [15:86].

Selection of the customer interface factor is sound, but requires some analysis of the inconsistent findings of the IBM-FSD and ITT studies. The ITT study indicates that, individually, increased user participation in requirements definition and increased customer experience both lead to increased productivity [22:146]. The IBM-FSD study shows that inverse relationships hold true [23:64]. Either relationship could be true, depending on the definition of the factor and the environment where it was measured. An intuitive example illustrates the point. A competent user with close participation in requirements definition would certainly increase productivity. On the other hand, a less competent user with high participation could interfere with productivity. The metric used to measure these factors isn't explained in either study. Therefore, the context of the results is unclear. What is clear from both studies is that customer relations contribute significantly to software productivity, and that a clear definition of its metric is essential.

FACTOR DEFINITION

Models that lack explicit definition of their constituent parts are of questionable value. A similar tone expressed in quotations from three authorities reveals the importance of clear definitions. First, the rigorous policy of the ITT study was that "Definitions were provided for each requested item of information to ensure consistent units of measure. ... The completed questionnaire required signature ... to guarantee that the data complied with the definitions" [22:144]. Second, Jones describes the state of software productivity estimating by saying that "lack of any standard definitions of those [common] metrics has introduced potential errors in excess of two orders of magnitude" [15:39]. Lastly, Conte believes that "A researcher should remember the definition of all metrics needed for a study" and that deviations from definitions should be fully explained "to maintain credibility of the research and the integrity of the results" [7:116]. The goal of establishing clear definitions for product complexity and customer relations factors is model credibility through consistency in estimating.

Product Complexity

Due primarily to its subjective nature, complexity is one of the most difficult factors to define. As a result, complexity metrics are a fertile field for research, and subjective complexity estimates are a fact of life in today's software industry [4:394-395]. This research doesn't attempt to eliminate subjective complexity estimates, but tries to increase their objectivity by defining specific attributes that characterize levels of complexity.

The COCOMO model has the best breakout of attributes contributing to complexity of any of the studies reviewed during this research. Its definition describes control, computational, device-dependent, and data management operations [4:391]. The other studies ignore the specific components and assign names to levels of complexity with unclear definitions. Although definitions lack rigor, all follow the same trend. That is, programs comprised primarily of simple mathematical computations have higher productivities than those requiring real-time techniques. Because COCOMO has the most comprehensive descriptions, it is the basis for defining the complexity factor for the SSC model. In fact, it appears to cover the multitude of complexity factors listed in the other studies [1:113; 4:118; 15:109; 22:146; 23:65].

Table 3, Product Complexity Ratings, defines the complexity factor by showing how the components of complexity contribute to a factor rating. The components of the table are sufficient to cover the Input Method, Output Method, Type System, and Data Base factors listed in Table 1, SSC Complexity Factor Breakout. The Privacy/Security factor doesn't appear significant and was omitted from the definitions. Such an attribute, as well as any other deemed meaningful in the future, can be easily added.

Customer Relations

The nature of customer relations is also very subjective. While authorities seem to agree that it has a significant impact on productivity, no one has attempted to define and study its impact. Since the task of the SSC is to write software systems to support USAF missions, their work is predominantly defined by organizations external to the SSC. Consequently, a workable definition is important in capturing the impact of customer relations on SSC productivity.

There are three aspects to customer relations that appear universally in the literature. One is how well the user understands both the specifics of the problem to be solved, and

Rating	Control Operations	Computational Operations	Device-dependent Operations	Data Management Operations
Very Low	Straightline code w/ a few non-nested ** SP operators: DOs, CASEs, IFTHENELSEs. Simple predicates.	Evaluation of simple expressions: e.g., A=B+C*(D-E).	Simple read, write statements w/ simple formats.	Simple arrays in main memory.
Low	Straightforward nesting of SP operators. Mostly simple predicates.	Evaluation of moderate-level expressions, e.g., D=SQRT(B**2-4*A*C).	No cognizance needed of particular processor or I/O device characteristics. I/O done at GET/PUT level. No cognizance of overlap.	Single file subsetting w/ no data structure changes, no edits, no intermediate files.
Nominal	Mostly simple math nesting. Some intermodule control. Decision tables.	Use of standard math + statistics routines. Basic matrix + vector operations.	I/O processing includes device selection, status check + error process.	Multifile INP + single file OUT. Simple structure changes, simple edits. Simple data bases.
High	Highly nested SP operators w/ many compound predicates. Queue + stack control. Considerable intermodule control.	Basic numerical analysis; multivariate interpolation, ordinary differential eq. Basic truncation, roundoff concerns.	Operations at physical I/O level (physical storage address translations; seeks, reads, etc). Optimized I/O overlap.	Special purpose subroutines activated by data stream contents. Complex data restructuring at record level. Complex data bases.
Very High	Reentrant + recursive coding. Fixed-priority interrupt handling.	Difficult but structured numerical analysis: near singular matrix equations, partial differential equations.	Routines for interrupt diagnosis, servicing, masking. Communication line handling.	Generalized, parameter-driven, file structuring routine. File building, command processing, search optimization.
Extra High	Multiple resource scheduling w/ dynamically changing priorities. Microcode level control.	Difficult + unstructured numerical analysis: highly accurate analysis of noisy, stochastic data.	Device timing-dependent coding, microprogrammed operations.	Highly coupled, dynamic relational structures. Natural language data management. Very complex data bases.

** SP = Structured Programming

Table 3. Product Complexity Ratings [4:391].

Rating	Problem Understanding	Requirements Definition	Lines of Communications
Low	General understanding of problem. Little functional experience. Little appreciation of automation capabilities.	Little clarification beyond function description. Initial requirements document not nailed down before design begins.	Infrequent. Low level of interest in development cycle reviews. Contacts limited to levels below those authorized to commit resources.
Nominal	Functional experience in all aspects of problem. Understands general scope of problem. Some appreciation for automation.	User assists in developing function description. FD sufficient to preclude most likely system modifications.	Lines of communication between responsible authorities open. Responsible representation at reviews.
High	Functional experience in all aspects of problem. Good understanding of automation limits.	User full time participation in developing comprehensive function description.	Proactive. Responsible authorities facilitate successful reviews. Timely, responsive correspondence.

Table 4. Customer Relations Ratings.

the contribution that automation can make in solving the problem. Another is how early in the life cycle that software requirements can be firmly defined. Third is how effective the lines of communications are between the customer and the software developer [4:488; 10:278-280; 22:146-147; 23:64]. Table 4, Customer Relations Ratings, categorizes these aspects into levels of productivity impact.

Unlike the complexity factor described above, a customer relations factor hasn't been adequately defined in the literature. Therefore, a brief discussion of its three aspects is illustrative. First, a customer's experience with the problem and understanding of the problem to be automated appear critical to deployment of a useful system. Software developers rely on functional experts to ensure that proposed software solutions satisfy USAF operational needs. Without competent functional expertise at critical points in the life cycle, late system changes are probable. This bears some relation to the second aspect of customer relations. That is, the better the customer understands their problem, the more likely it is that requirements can be frozen early, where they have less negative impact on productivity. Failure to nail down requirements leads to continual changes in areas such as requirements, personnel, facilities, and resources, even as the development cycle progresses. The third aspect, open communications, isn't only highly correlated with the first two, but is critical throughout the development cycle. Open communications tend to reduce misunderstandings that lead to frequent project redirections. In summary, the ideal customer relationship is one where functionally competent users nail down their requirements early and stay involved throughout the development cycle.

FACTOR INITIALIZATION

To reiterate the initialization problem, there is no history upon which to base initial factor values. Under this reality, theory becomes the basis for establishing factor values. While these theoretical values will ultimately require calibration, they are the first step towards reliable software cost estimates at the SSC. This section justifies and assigns initial values to the product complexity and customer relations factors.

Product Complexity

Three studies and two product evaluations provide support for assigning the COCOMO values, unchanged, to the six levels of product complexity outlined above. The studies by IBM-FSD, ITT, and COCOMO are outlined under Factor Selection above. The evaluations, performed in predominantly COBOL environments, listed the following cautious conclusions related to complexity. One found "no evidence that the COCOMO effort multipliers [including product complexity] are invalid" [17:298]. The other indicated that "it is possible that COCOMO is accurately reflecting the project conditions, but is calibrated too high" [16:422].

Without a history to the contrary, the COCOMO values intuitively appear to be good initial values, especially when compared to the IBM-FSD and ITT studies. Table 5 shows multiplier values for categories obtained from the IBM-FSD study as related by Humphrey (Note: These values weren't published in the original Walson-Felix study, but were obtained by Humphrey via an interview with a person presumably familiar with the study) [12:5,22]. The ITT and COCOMO values listed in the table are derived by applying their respective complexity values to the IBM-FSD categories. On a gross level, each set of values was derived from a fair sized set of projects and a wide variety of applications [4:83; 22:144; 23:57]. Notice that the COCOMO values fall between the others. This fact motivates selection of the COCOMO values to initialize the product complexity factor. Selecting a viable middle ground is a hedge against outrageous estimates until data can be gathered to calibrate them.

Category	IBM-FSD	COCOMO	ITT
Concurrent Processes	2.5	1.7	1.3
Interface Software	1.7	1.3	
Complex Logic	1.3	1.2	
Math Function	1.0	.9	.5

Table 5. Comparison of Complexity Multipliers

The hypothesis that they are viable is supported by two COCOMO model evaluations. Both Miyazaki and Kemerer found that the COCOMO model tended to overestimate effort. However, both laid primary blame on the baseline equation and gave limited support to the validity of the effort multipliers. Interestingly, Miyazaki stated there was no evidence of invalidity of the multipliers, and was using 12 of the 15, including complexity, unchanged. Kemerer found that, contrary to their intent, COCOMO baseline estimates were better than those of the adjusted baseline. A contributing factor could be Kemerer's decision not to weight COBOL nonprocedural statements by one third as suggested by COCOMO. Without this reduction applied to the baseline, multipliers greater than one will aggravate the overestimate. Still, Kemerer suggests that the COCOMO multipliers may be valid. Regardless, both evaluations suggest

that the reliable use of COCOMO values in other environments requires their calibration [16:421-423; 17:295-299].

As a result of this analysis, the following values are taken directly from COCOMO and assigned to the rating categories in Table 3, Product Complexity Ratings: Very Low - .7, Low - .85, Nominal - 1.0, High - 1.15, Very High - 1.3, and Extra High- 1.65 [4:118].

Customer Relations

The values assigned to customer relations are essentially those identified in the ITT study, with minor modifications. The values are: low - 1.6, nominal - 1.0, and high - .6. These values are the product of the means of the client participation and experience attributes as interpreted from the graphs in the study. While these factors aren't independent, their correlation isn't factored into the values because supporting data wasn't available in the published article [22:146-147]. Still, the ITT values are within the range of the potential cost doubling predicted by Boehm [4:488], and are the initial values suggested by this research. As with product complexity, they require calibration at the earliest possible time.

Chapter 4

APPLYING THE PRODUCT FACTORS

In the previous chapter, two product factors were shown to have potentially significant effects on software productivity at the SSC. Product complexity and customer relations factors were selected, defined, and initialized based on their contributions to productivity as analyzed by authoritative studies. This chapter briefly describes how to integrate these factors into the SSC model, using a top-down approach. That is, first, the method for using the factors to adjust the baseline estimate is presented, followed by a description of how to determine the factor values, themselves. Somewhat independently of the top-down explanation, the last section of this chapter discusses future calibration of the factors.

The reason for a top-down explanation is that product complexity and customer relations factors adjust the baseline differently. Understanding how they apply facilitates discussion of how each is determined. Product complexity is applied to individual modules of the WBS. So, the categories of product complexity apply to manageable sublevels of the problem. On the other hand, the categories that describe customer relations generally apply to the system as a whole, affecting each element of the WBS equally. So, the first task of this chapter is to take a macro view of the relationship between the factors and the SSC model by describing how the product factors adjust the baseline estimates.

ADJUSTING THE BASELINE

Factor values are used as effort multipliers of baseline estimates and have an inverse relationship to productivity. That is, values greater than one increase the equivalent number of SLOC required to complete a job, thereby effectively decreasing productivity. Conversely, values less than one decrease equivalent SLOC and increase productivity estimates. While actual product SLOC doesn't change as a function of product factors, productivity does. Thus product factors increase and decrease equivalent SLOC, but don't affect actual SLOC.

Product Complexity

Product complexity, as defined in Chapter 3, isn't constant across the entire system and may not even be constant within a module of the WBS. That is, half of the modules of the system could be simple math procedures, while the other half could be difficult numerical analysis routines. Applying only one or the other factor to the entire system would grossly skew the resulting productivity estimate. But, applying the proper factors to system components and summing the results gives a more accurate estimate.

Product complexity applies to the lowest levels of the WBS. It is at these levels that module definition is explicit enough to reliably estimate the SLOC and to characterize the structures, required to implement the function. After product complexity factors are determined (see following section), they are multiplied by the estimated SLOC count for the appropriate WBS element to yield an equivalent SLOC estimate. Then the equivalent SLOC estimates are summed to obtain an equivalent estimate for the total system.

For example, suppose that a problem to be automated requires a system of three modules. The first module is estimated to need 2300 SLOC and has a complexity rating of .8. The second and third require 4000 and 1800 SLOC, and have ratings of 1.65 and 1.15, respectively. The modules require the following equivalent SLOC: first - 1840, second - 6600, and third - 2070. The total system will actually consist of 8100 SLOC, but will require the effort of 10,510 to implement.

Note that the term - reliable - as used in "reliably estimate the SLOC" in the second paragraph of this subsection, connotes relative significance. The degree of reliability attainable depends on the level of understanding of the problem to be automated and the phase of development where the estimate is made. The relative nature of reliable estimates is accounted for by a contingency factor in the SSC model, not by product factors.

Customer Relations

Unlike product complexity, the customer relations factor applies to groups of modules within the WBS. In the simplest case a single value applies to the system as a whole. In general, the three aspects of customer relations - problem understanding, requirements definition, and lines of communications - individually, apply at the same rating level across all modules within the system. That is, if lines of communications are open between the customer and the SSC, then they will be open for all elements of the WBS. Therefore, the customer relations factor is applied at the system level,

immediately following application of the product complexity factor.

Continuing with the example above, assume that relations with the customer are excellent and result in a value of .6. This value is used to scale the equivalent SLOC estimate of 10,510 to yield 6300. Again, the system will actually consist of 8100 SLOC, but now the effort has been scaled to 6300 to account for customer relations.

The scenario outlined above is a simple case. In practice, customer relations may vary with elements of the WBS. Consider an example where a system consists of two major subsystems, and the customer is very familiar with one subsystem and vaguely familiar with the other. Intuitively, development of the familiar subsystem will be more productive than development of the other. To accurately reflect this situation, different values must be applied to each of the subsystems.

In summary, customer complexity values apply to groups of modules within the WBS. In the simplest case one value is sufficient to accurately scale the entire system. In more complex cases more than one may be required. Consequently, it is incumbent upon the software estimator to determine the scope of effect for each value and apply it accordingly.

DETERMINING THE FACTOR VALUES

The section above described how to apply product factors to the baseline estimate. This section takes a microscopic view of how each of these product factor values is determined.

Product Complexity

To obtain a product complexity value for a module, it must first be characterized according to the levels defined in Table 3, Product Complexity Ratings. This is done by identifying the predominant characteristic of a module and assigning the corresponding rating to the module. Note that a module won't exhibit characteristics from each of the four types of operations. When determining a rating, those types which don't apply to the predominant module characteristic should be ignored. Once a rating is determined, it is translated into a complexity value according to the following scale: very low - .7, low- .85, nominal - 1.0, high - 1.15, very high - 1.3, and extra high - 1.65. For example, a module characterized by highly nested structured programming operators with many compound predicates, and considerable intermodule control, receives a complexity value of 1.15.

Ideally, a module will exhibit singularly well-defined characteristics. Such cases follow the procedures outlined above. Unfortunately, some modules can't be categorized so simply. Their characteristics may be significantly split between types and levels of operators, so as to invalidate a value based on a single characteristic. Consider a module characterized by very low and extra high data management operations and nominal computational operations. Further, suppose that the percentage of SLOC for each type is 20, 40, and 40, respectively. Then a value for the module can be calculated by the sum of the values contributed by each level. That is, the product complexity value for this example is (.2 * .7) + (.4 * 1.65) + (.4 * 1.0) = 1.2.

Customer Relations

Customer relations values are obtained similarly to product complexity in that ratings derived from a table are translated into factor values. But, the method is very different. Every system under development exhibits traits from all three aspects of customer relations, namely: problem understanding, requirements definition, and lines of communications. Any value assigned must account for all three. This is accomplished by obtaining an average value to describe customer relations. That is, the system is rated on each of the three aspects and is assigned an associated value from: low - 1.6, nominal - 1.0, or high - .6. The three values are then added together and divided by three to obtain the customer relations multiplier.

CALIBRATING THE FACTORS [4:377-379]

As stated previously, theoretical values used in a model must be calibrated to the environment where they will be used. Calibration in the case of the SSC model includes the baseline and all of its associated factors. The method for adjusting the product factors presented in this section assumes that the baseline equation is calibrated. Without a reasonably stable baseline, adjustment factors can't be reliably calibrated. This is a consequence of the fact that the baseline defines the magnitude of the development effort, while adjustments scale the effort based on process and product variables. The implication follows that uncalibrated baseline estimates potentially contribute more to estimating error than individual adjustment factors, effectively obscuring the factors.

Boehm provides a method for analyzing factor behavior called ideal effort multiplier (IEM). In the SSC case for product factor analysis, the IEM is a function of the project (P) and the product factor (PF) being analyzed, and is denoted IEM(P,PF). More specifically, IEM(P,PF) is equal to the number of man-months required to actually produce the system divided by the estimated

man-months for the same project, excluding the PF being analyzed. This ratio yields the percentage of error that can't be explained by all of the other factors used in the estimate. By implication the unexplained error is due to the PF being analyzed. Further, IEM(P,PF) is the multiplier that causes an individual project estimate to equal actual project effort.

Mathematically, the relationship between IEM(P,PF), estimated effort, and actual effort is easy to see. Let MM(P,act) equal the actual number of man-months needed to complete project P, and MM(P,PF) equal the estimated man-months to complete the same project, excluding the PF of interest. By definition, IEM(P,PF) = MM(P,act) / MM(P,PF). So, MM(P,act) = IEM(P,PF) * MM(P,PF), showing the relationship identified in the previous paragraph.

Overall, the goal in IEM analysis is to identify a factor value that best describes the variance contributed by that factor via comparing actual project data with estimates involving the factor. This is accomplished with the following algorithm:

1. Factor the PF of interest out of the project estimates. For product complexity this probably requires recalculating estimates without including complexity factors. In the case of customer relations, this can be done effectively by dividing each project estimate by its associated factor value.

2. Calculate IEM(P,PF) for each project using the values from step 1 in the denominator. This yields a series of multipliers that make individual estimates equal actual values.

3. Plot the IEM(P,PF) against the SSC multiplier for all projects. Graphically, this illustrates how well each SSC value matches the ideal values of IEM(P,PF).

4. Perform a regression analysis of the resulting data to identify more accurate values for the product factor being analyzed.

The IEM algorithm analyzes a single factor at a time by correlating variance from actual values to a particular factor. This technique implies that factors other than the one being analyzed accurately reflect their related variances, and any remaining variance is totally attributed to the factor in question. Considering the level of maturity of the SSC model, neither implication is accurate. To compensate for this weakness, the IEM algorithm must be executed for each of the product factors. Then the entire process is reiterated, using the values obtained in the previous iteration, until factor values converge on an acceptable solution.

Chapter 5

CONCLUSIONS AND RECOMMENDATIONS

The need for a reliable software cost estimating model at the SSC is well-established. The phenomenal growth in the importance of software engineering economics dictates a non-trivial solution. No cost models, either proprietary or non-proprietary, exist for general use in a particular environment. All models require some level of customizing. The SSC chose to implement a non-proprietary model based on the theories of Humphrey. This research project was commissioned to customize the model to account for product induced adjustments, which applied in the SSC environment.

FACTOR EVALUATION BASED ON SELECTION CRITERIA

Due to a lack of significant historical data, the job of customizing the model became a theoretical selection process to identify, define, and initialize a set of relevant product factors. But first, criteria to guide the selection of product factors were established. The five criteria - significance, scope, ease of use, measurability, and factor independence - were introduced at the end of Chapter 2, just prior to initiating the selection process. Fittingly, the last task in the process is evaluating the chosen factors against the selection criteria. The following evaluations carry the caveat that this research was predominantly theoretical. Only history can prove the goodness of the criteria and whether the chosen factors meet them.

Significance

The significance criterion established the need for two product factors with PRs greater than 200 percent. Two factors overshadowed all others in the study and were selected for application at the SSC. Both product complexity and customer relations had PRs greater than 2.0 in the studies where they were assigned values. Further, the initial values assigned by this research produces PRs of 2.4 for product complexity and 2.7 for customer relations.

Scope

There are three aspects of the scope of this research project. First, solutions apply only to software production and modification. Second, factors account for the definition, design, development, and test phases of the life cycle. Finally, SLOC counting includes only delivered SLOC that are input to a compiler, excluding comment and blank lines. The studies used in this research each have different definitions for each of these aspects. But, the COCOMO and ITT studies were the primary sources for defining and initializing the product complexity and customer relations factors.

Software maintenance is sufficiently different from new development and modification that it requires a unique cost estimating procedure. The COCOMO states directly that it is valid only for production and modification functions, while the ITT study appears to imply the same. Use of the two studies satisfy the criterion for factor use in production and modification environments [4:59; 22:143-145].

Both studies indicate that cost estimates include life cycle activities from development through testing. It isn't clear from the ITT definition of scope whether the software definition phase is included or not. The COCOMO study explicitly omits the definition phase. Of the two factors, customer relations is the one most affected by activities in the definition phase. Considering the inexplicit scope of the ITT study, the customer relations factor bears close scrutiny for early calibration. Also, omission of the definition phase has little impact on the product complexity factor as defined in this research. So, the life cycle criterion is adequately covered by the studies to allow initialization of product factors [4:59; 22:145].

While the criterion and the two studies all count SLOC similarly, there are minor differences owing primarily to the robustness of the definitions in the studies. For example, COCOMO includes job control language in its count, and ITT counts the statements generated by a macro call. The scope criterion isn't so explicit. Again, the definitions are sufficiently close to allow the studies to be a basis for initial factor values [4:58-59; 22:145].

Ease of Use

Ease of use is exemplified by a factor that is significant, well-defined, and easy to obtain. The significance of the product factors is discussed two subsections above. The factors are well-defined in that category definitions cover a broad, comprehensive, and clearly defined spectrum of factor attributes. Finally, software characteristics used to quantify the factors

are easy to obtain from the perspective of the estimating model in that they are a byproduct of a normal design activities.

Measurability

Measurability encompasses the concepts of objectivity and quantitativeness. The table lookup procedures for assigning factor values suggested by this research meet the need for quantifiable factors. On the other hand, they don't provide for total objectivity. Estimating future project attributes and the intents of people will perpetually involve some level of subjective judgement. But, subjectivity is minimized by classifying WBS modules according to well-defined categories of software attributes.

Factor Independence

The category definitions for the product factors are intuitively independent. Product complexity involves software characteristics, while customer relations involves interpersonal relationships. It may be possible to devise a scenario where software characteristics and interpersonal relationships can't be effectively separated. However, the factors don't generally appear to be correlated.

RECOMMENDATIONS

This report concludes with two recommendations. One is that the customer relations factor be refined before being put into use. The second is that the two product factors be incorporated into the current SSC software cost estimating model.

Of the two factors chosen by this research, customer relations is the most immature. While product complexity is explicitly defined and used in COCOMO, customer relations is discussed only generically in the literature. As a result, the definitions of customer relations listed in this research are original and haven't been validated in an operational environment. This leads to the recommendation that the customer relations factor be refined by functional experts prior to its use at the SSC. This can best be accomplished through the Wideband Delphi technique espoused by Boehm [4:333-335] and taught at the SSC by the SEI [19:4-1 - 4-6].

The second recommendation is to integrate the product complexity and customer relations factors into the SSC model. Authoritative studies have consistently demonstrated that productivity factors influence software costs. Further, the recommended method for customizing factors to a specific environment is by applying history. But, available history is

insufficient for the task at the SSC. So, theoretical factors must suffice until empirical evidence indicates better alternatives. The two factors identified by this research are reasonable starting points.

Integration of the product complexity and customer relations factors implies more than merely directing managers to use the factors. It also requires a disciplined history recording program. The perception that cost estimates based on theoretical factors are less credible than those with a historical foundation, heightens the need for empirical calibration. But, data must be carefully regulated and documented to be useful. Inconsistent data collection leads to inconclusive analysis, or worse, erroneous conclusions. A mature, credible software cost estimating model can only evolve through the application of consistent data.

In conclusion, the old adage attributed to Lao-tzu is apropos, "A journey of a thousand miles must begin with a single step." Factor usage based primarily on theory isn't the preferred way to obtain software cost estimates. But, it is an acceptable first step in a long journey towards reliable cost estimates.

BIBLIOGRAPHY

1. Bailey, John W. and Victor R. Basili. "A Meta-model for Software Development Resource Expenditures." Fifth International Conference on Software Engineering, Wash DC: IEEE Computer Society Press, 1981.

2. Basili, Victor R. "Resource Models." Tutorial on Models and Metrics for Software Management and Engineering, Wash DC: IEEE Computer Society Press, 1980.

3. Boehm, Barry W. "Software Engineering." Classics in Software Engineering, edited by E. N. Yourdon, New York: Yourdon Press, 1979.

4. Boehm, Barry W. Software Engineering Economics. Englewood Cliffs, NJ: Prentice-Hall Inc, 1981.

5. Brock, Patrick E., TSgt, USAF. SSC CAMPUS Administrator, HQ Standard Systems Center (SSBSM) (Personal Conversations and Database Queries). Gunter AFS, AL.

6. Brooks, Fredrick P. The Mythical Man-Month. Reading, MA: Addison-Wesley Publishing Company, 1975.

7. Conte, Samuel D., H. E. Dunsmore and V. Y. Shen. Software Engineering Metrics and Models. Benjamin/Cummings Publishing Company, 1986.

8. Dettl, Mark, Capt, USAF. SSC Cost Model Point of Contact, HQ Standard Systems Center (PRS) (Personal Conversations). Gunter AFS, AL.

9. Dunham, Janet R. and Elizabeth Kruesi. "The Measurement Task Area." Computer, November 1983, pp. 47-54.

10. Evans, Michael W. and John J. Marciniak. Software Quality Assurance and Management. New York: John Wiley and Sons, 1986.

11. Goel, Amrit L. "COMPSAC 83 Panel Session on Data Analysis and Modelling in Software Engineering: Theory and Practice." The IEEE Computer Society's Seventh International Computer Software and Applications Conference, Wash DC: IEEE Computer Society Press, 1983.

12. Hayes, John P. Computer Architecture and Organization. New York: McGraw-Hill Book Company, 1978.

13. Humphrey, W. S. "Extract from Software Process Management." Software Engineering Institute, Carnegie-Mellon University, Pittsburgh, PA, 1987.

14. Jones, Capers. "Introduction." Tutorial on Programming Productivity: Issues for the Eighties, Wash DC: IEEE Computer Society Press, 1986.

15. Jones, Capers. Programming Productivity. New York: McGraw-Hill Book Company, 1986.

16. Kemerer, Chris F. "An Empirical Validation of Software Cost Estimation Models." Communications of the ACM, May 1987, pp. 416-429.

17. Miyazaki, Yukio and Kuniaki Mori. "COCOMO Evaluation and Tailoring." Eighth International Conference on Software Engineering, Wash DC: IEEE Computer Society Press, 1985.

18. Software Engineering Institute. "Final Report and Recommendations, Software Engineering Institute Review, Standard Systems Center, Gunter Air Force Station. 17-19 Feb 87." Carnegie-Mellon University, Pittsburgh, PA, 1987.

19. Software Engineering Institute. "Participants Guide: Software Project Planning Course, version 0.9." Carnegie-Mellon University, Pittsburgh, PA, 1987. Course taught at HQ Standard Systems Center (PRS), Gunter AFS, AL, September 1987.

20. US Department of the Air Force: HQ Standard Systems Center (PRS). Response to Software Engineering Institute Findings and Recommendations. Gunter AFS, AL, 7 Aug 87.

21. US Department of the Air Force: HQ Standard Systems Center (SSB). SSCR 700-7, Vol. I: Center Automated Manpower and Project Update System. Gunter AFS, AL, 15 Oct 87.

22. Vosburgh, J., et. al. "Productivity Factors and Programming Environments." Seventh International Conference on Software Engineering, Wash DC: IEEE Computer Society Press, 1984.

23. Walston, C. E. and C. P. Felix. "A Method of Programming Measurement and Estimation." IBM Systems Journal, Vol. 16, No. 1, 1977, pp. 54-73.

END

DATE

FILMED

6-88

DTIC